On Routing and Spectrum Assignment in Rings

Sahar Talebi, Evripidis Bampis, Giorgio Lucarelli, Iyad Katib, and George N. Rouskas

***Abstract*—We present a theoretical study of the routing and spectrum assignment (RSA) problem in ring networks. We first show that the RSA problem with fixed-alternate routing in general-topology (mesh) networks (and, hence, in rings as well) is a special case of a multiprocessor scheduling problem. We then consider bidirectional ring networks and investigate two problems: 1) the spectrum assignment problem under the assumption that each demand is routed along a single fixed path (e.g., the shortest path), and 2) the general case of the RSA problem whereby a routing decision along the clockwise and counter-clockwise directions must be made jointly with spectrum allocation. Based on insights from multiprocessor scheduling theory, we derive the complexity of the two problems and develop new constant-ratio approximation algorithms with a ratio that is strictly smaller than the best known ratio to date.**

***Index Terms*—Approximation algorithms, multiprocessor scheduling, optical fiber networks, routing and spectrum assignment, routing and wavelength assignment, spectrum assignment, wavelength assignment.**

I. Introduction

ELASTIC optical networking has been the subject of considerable research and development activities in recent years due to its potential to accommodate efficiently the ongoing growth in traffic demands [1]–[3]. Key enabling technologies of elastic networking include optical OFDM, distance-adaptive modulation, flexible spectrum selective switches, and bandwidth-variable transponders [4]. These technologies make it possible for network operators to support multirate connections and adapt to variable bandwidth requests dynamically, by "slicing off" just the right amount of spectrum for each traffic demand [2].

Routing and spectrum assignment (RSA) [5]–[7] has emerged as the essential problem for network-wide management of spectral resources in the context of design and control of elastic optical networks. The objective of the RSA problem is to (1) assign a physical path to each demand, and (2) allocate continuous and contiguous spectrum to the demand along the links of each path, so as to optimize a metric of interest typically related to spectrum utilization. Several offline and online variants of the problem have been studied; for a survey and classification of existing approaches, the reader is referred to [8]. Since general versions of the RSA problem are computationally intractable, common solution approaches include integer linear programming formulations (for small network sizes) and heuristics.

While many studies of the RSA problem consider general network topologies, we note that in addition to the fact that large parts of the current infrastructure are based on SONET/SDH rings, DWDM networks with topological rings are being deployed based on technologies other than SONET (e.g., Ethernet, IP/MPLS, etc.), including for wireless backhaul to accommodate the explosive growth of mobile data [9]–[12]. Therefore, more recently there has been increasing interest in RSA solutions for ring networks [13]–[17]. Most of these studies employ heuristics, although some interesting theoretical results do exist. For instance, using results from graph coloring theory, it was shown in [17] that there exists a $(4+2\epsilon)$-approximation algorithm for the spectrum assignment (SA) problem in rings; whereas the work in [16] proves that the contiguity (i.e., adjacency) constraint in SA can always be satisfied starting from an optimal solution to a corresponding coloring problem.

In this work, we present a comprehensive study of the RSA problem in bidirectional ring networks, and make three contributions. In Section II, we show that the RSA problem with fixed-alternate routing in general-topology (mesh) networks is a special case of a multiprocessor scheduling problem. Building upon this new perspective, we investigate theoretically two approaches to tackling the RSA problem in rings. In Section III, we consider the case of a single fixed path for each demand; this corresponds to a two-step approach for RSA in which routing of demands is performed first, followed by spectrum allocation. When each traffic demand follows a predetermined path (e.g., the shortest path) from source to destination, RSA reduces to the SA problem. We prove that the SA problem can be solved in polynomial time in small bidirectional rings, and we develop constant-ratio approximation algorithms for large rings. In Section IV, we show that the RSA problem, in which routing and spectrum allocation are considered jointly, is intractable for rings with as few as four nodes. Based on insight from multiprocessor scheduling theory, we also develop an approximation algorithm for ring RSA. The approximation ratios of our algorithms are strictly smaller than the best known $(4+2\epsilon)$ ratio presented in [17]. We also note that our results apply to the routing and wavelength assignment problem, a special case of

Manuscript received August 4, 2014; revised October 31, 2014; accepted November 20, 2014. Date of publication December 1, 2014; date of current version January 23, 2015. This work was supported in part by the National Science Foundation under Grant CNS-1113191 and by the Deanship of Scientific Research, King Abdulaziz University, under Grant 2-611-1434-HiCi.

S. Talebi is with the Operations Research and Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206 USA (e-mail: stalebi@ncsu.edu).

E. Bampis and G. Lucarelli are with the UPMC Univ Paris 06, Sorbonne Universités, F-75005 Paris, France (e-mail: bampis@gmail.com; glucarelli@gmail.com).

I. Katib is with the King Abdulaziz University, Jeddah 21441, Saudi Arabia (e-mail: iakatib@kau.edu.sa).

G. N. Rouskas is with the Operations Research and Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206 USA, and also with the King Abdulaziz University, Jeddah 21441, Saudi Arabia (e-mail: rouskas@ncsu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JLT.2014.2376871

RSA in which all demands are of equal size. We conclude the paper in Section V.

II. RSA in General Topology Networks: A Special Case of Multiprocessor Scheduling

Consider the following general definition of the RSA problem with fixed-alternate routing in elastic optical networks; in this definition, we assume that a spectrum slot is the smallest amount of spectrum (e.g., 12.5 or 6.25 GHz, depending on the technology), that can be allocated and routed as a single unit.

Definition II.1 (RSA): Given

- a graph $G = (\mathcal{V}, \mathcal{A})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{A}$ the set of arcs (directed links),
- a spectrum demand matrix $T = [t_{sd}]$, where t_{sd} is the number of spectrum slots required to carry the traffic from source node s to destination node d, and
- k alternate routes, $r^1_{sd}, \ldots, r^k_{sd}$, from node s to node d,

assign a route and spectrum slots to each demand so as to minimize the total amount of spectrum used on any link in the network, under three constraints:

1) each demand is assigned contiguous spectrum slots (spectrum contiguity constraint);
2) each demand is assigned the same spectrum slots along all links of its path (spectrum continuity constraint); and
3) demands that share a link are assigned non-overlapping parts of the available spectrum (non-overlapping spectrum constraint).

If each traffic demand is constrained to follow a specific route from source to destination that is provided as part of the input, (i.e., $k = 1$ in the above definition,) then the RSA problem reduces to the SA problem.

In recent work [18], we have shown that the SA problem in (mesh) networks of general topology is a special case of the classical multiprocessor scheduling problem denoted as $P|fix_j|C_{max}$. In other words, every instance of the SA problem can be transformed to an instance of $P|fix_j|C_{max}$ (whereas the reverse is not true), and hence, any algorithm that solves $P|fix_j|C_{max}$ also solves SA. Problem $P|fix_j|C_{max}$, in which tasks are to be executed on multiple processors simultaneously, is formally defined as [19], [20]:

Definition II.2 ($P|fix_j|C_{max}$): Given

- a set of m identical processors,
- a set of n tasks with processing time $p_j, j = 1, \ldots, n$, and
- a prespecified set fix_j of processors for executing each task $j, j = 1, \ldots, n$,

schedule the tasks so as to minimize the makespan $C_{max} = \max_j C_j$, where C_j denotes the completion time of task j, under the constraints:

1) preemptions are not allowed;
2) each task must be processed simultaneously by all processors in set fix_j; and
3) each processor can work on at most one task at a time.

Also, we denote by $Pm|fix_j|C_{max}$ the special case of $P|fix_j|C_{max}$ in which the number of processors m is considered to be fixed. The proof of the transformation is available in [18]. Briefly, each link in the SA problem transforms to a processor, each traffic demand (s, d) to a task j, the demand size t_{sd} and path r_{sd} to the processing time p_j and set fix_j of the corresponding task j, respectively, the maximum spectrum assigned to any link to C_{max}, and each of the three constraints of the SA problem to the similarly numbered constraint of problem $P|fix_j|C_{max}$.

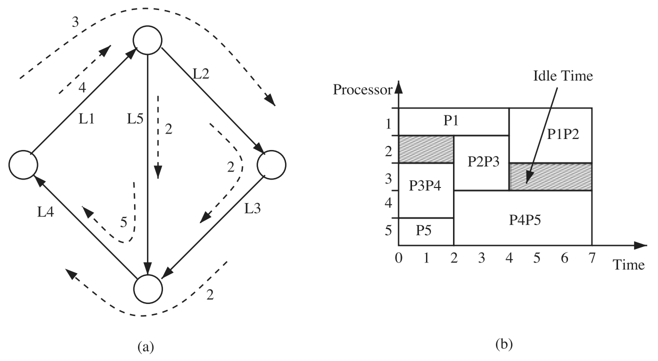

Fig. 1. (a) Instance of the SA problem on a mesh network with five directed links (arcs). (b) Optimal schedule of the corresponding $P|fix_j|C_{max}$ problem.

As an example, Fig. 1(a) shows an instance of the SA problem on a mesh network with five directed links, $L1, L2, L3, L4$, and $L5$. There are five demands, shown as dotted lines, with the number of slots required by each demand shown next to the corresponding line. Fig. 1(b) shows the optimal schedule for the $P|fix_j|C_{max}$ problem corresponding to this SA instance, whereby link $L1$ maps to processor $P1$, link $L2$ to processor $P2$, and so on. As we can see, the demand of size 3 that follows the path $L1$-$L2$ is mapped to a task that is scheduled in the time interval $[4, 7]$ on the corresponding processors $P1$ and $P2$; similarly for the other demands. The schedule is optimal in that $C_{max} = 7$ is equal to the total processing time required for processors $P1, P4$ and $P5$. Also, the value of C_{max} is equal to the total number of spectrum slots required for links $L1, L4$, and $L5$.

We now show that the RSA problem with fixed-alternate routing in networks of general topology can also be viewed as a multiprocessor scheduling problem. Consider the following scheduling problem that has been studied in the literature [21]–[23]:

Definition II.3 ($P|set_j|C_{max}$): Given

- a set of m identical processors,
- a set of n tasks with processing time[1] $p_j, j = 1, \ldots, n$,
- and a prespecified set $set_j = \{fix^1_j, \ldots, fix^k_j\}$ of k alternative processor sets to which task j can be assigned,

schedule the tasks so as to minimize the makespan $C_{max} = \max_j C_j$, where C_j denotes the completion time of task j, under the constraints:

[1]In its most general form, the $P|set_j|C_{max}$ problem allows for the processing time p^i_j of a task j to depend on the processor set fix^i_j to which the task is assigned. This general variant of the problem may be used to model distance-adaptive RSA, in which the modulation format, and hence, the required amount of spectrum, depends on the length of the path assigned to the demand. Distance-adaptive RSA is outside the scope of this paper, and is the subject of ongoing research in our group; therefore, we only consider the version of the scheduling problem in which a task's processing time is independent of the processor set to which it is assigned.

1) preemptions are not allowed,
2) each task must be processed simultaneously by all processors in only one of the processor sets in set_j, and
3) each processor can work on at most one task at a time.

Clearly, $P|fix_j|C_{max}$ is a special case of $P|set_j|C_{max}$ with $k = 1$ processor set for each task. It has been shown that, in the general case, there can be no constant-ratio polynomial time approximation algorithm for $P|set_j|C_{max}$ unless $P = NP$ [24]. The two-processor problem $P2|set_j|Cmax$ has been proven in [25] to be NP-hard. Approximation algorithms with ratio m and $m/2$ have been developed in [21] and [22], respectively, for problem $Pm|set_j|C_{max}$, in which the number m of processors is considered to be fixed and is not part of the input (as it is for $P|set_j|C_{max}$). Also, polynomial time approximation schemes (PTAS) for $Pm|set_j|C_{max}$ were introduced in [23], [26].

The following two lemmas show that the RSA problem with fixed-alternate routing in mesh networks is a special case of $P|set_j|C_{max}$.

Lemma II.1: RSA with fixed-alternate routing in mesh networks transforms to $P|set_j|C_{max}$.

Proof: Consider an instance of RSA with fixed-alternate routing on a network with a general topology graph $G = (\mathcal{V}, \mathcal{A})$, demand matrix $T = [t_{sd}]$ and set $\{r^1_{sd}, \ldots, r^k_{sd}\}$ of alternate routes for source-destination pair (s, d). Construct an instance of $P|set_j|C_{max}$ such that:

- for each directed arc $a_l \in \mathcal{A}$, there is a processor l, and
- for each spectrum demand t_{sd}, there is a task j with $p_j = t_{sd}$ and $set_j = \{fix^1_j, \ldots, fix^k_j\}$, where $fix^i_j = \{q : a_q \in \{r^i_{sd}\}\}$.

Hence, the amount of spectrum of a demand transforms to the processing time of the corresponding task, the set of alternate paths to the set of alternate processor sets for that task, and the links of each alternate path to the corresponding set of alternate processors for the task. Due to the spectrum contiguity constraint, preemptions are not allowed. The spectrum continuity constraint guarantees that each task will be processed simultaneously by all processors in the alternate set assigned to the task, whereas the non-overlapping spectrum constraint requires that each processor work on at most one task at a time.

By construction, the amount of spectrum assigned to any arc of G in a solution of the RSA instance is equal to the completion time of the last task scheduled on the corresponding processor, hence minimizing the spectrum on any link in the RSA problem is equivalent to minimizing the makespan of the schedule in the corresponding problem $P|set_j|C_{max}$. ■

We now show that the reverse of Lemma II.1 is not true. In other words, there exist instances of $P|set_j|C_{max}$ for which there is no corresponding instance of the RSA problem.

Lemma II.2: There exist instances of $P|set_j|C_{max}$ for which there is no corresponding instance of the RSA problem.

Proof: By counterexample. Consider an instance of $P|set_j|C_{max}$ with eight processors $\{1, 2, 3, 4, 1', 2', 3', 4'\}$, and these five tasks whose processing times can be arbitrary:

task	fix^1_j	fix^2_j
τ_1	$\{1, 2\}$	$\{4', 3'\}$
τ_2	$\{2, 3\}$	$\{1', 4'\}$
τ_3	$\{3, 4\}$	$\{2', 1'\}$
τ_4	$\{4, 1\}$	$\{3', 2'\}$
τ_5	$\{2, 4\}$	$\{4', 2'\}$.

Because of the first four tasks, the graph of the corresponding RSA instance would have to be the four-node, eight-link bidirectional ring network such that: (1) links l and l', $l = 1, 2, 3, 4$, are links in the clockwise and counter-clockwise direction, respectively, between adjacent nodes in the ring, (2) in the clockwise direction, link 1 is adjacent to 2, 2 is adjacent to 3, 3 to 4, and 4 to 1, and (3) similarly for links in the counter-clockwise direction. Since there are no feasible paths for the spectrum demand corresponding to the last task τ_5, an instance of RSA does not exist. ■

III. SA in Rings

In this section, we study the SA problem in bidirectional rings under the assumption that each traffic demand is carried over the shortest path from its source to the destination node; we defer discussion of the RSA problem, in which the RSA problems are solved jointly, to the next section. Let N be the number of nodes of the ring network. Note that, whenever N is even, there are two shortest paths between every pair of nodes that are diametrically opposite each other. In this case, we assume that one of these paths (in either the clockwise or counter-clockwise direction) is selected and is provided as input to the SA problem.

We first note that, under shortest path routing, the clockwise and counter-clockwise directions of the ring become decoupled and completely independent of each other. Consequently, the SA problem in bidirectional rings is decomposed into two disjoint subproblems, one for each direction, that can be solved separately; the subproblem in the clockwise (respectively, counter-clockwise) direction takes as input the subset of clockwise (respectively, counter-clockwise) links and the subset of demands with shortest paths along these links. It can be seen that this decomposition is optimal, in that finding the optimal solution (i.e., minimum total spectrum on any link) for each subproblem and taking the maximum of the two is an optimal solution to the original problem on the bidirectional ring. Therefore, for the remainder of this paper, we will only consider the SA subproblem for the clockwise direction of the ring. Because of symmetry, the same results apply to the subproblem defined on the counter-clockwise direction, although the optimal solution may be different (e.g., because of the fact that different demands are placed on each direction).

We have shown in [18] that the SA problem in *unidirectional rings* can be transformed to a $P|fix_j|C_{max}$ problem. Moreover, in the general case, i.e., whenever there are traffic demands between any pair of nodes, the SA problem in unidirectional rings with $N = 3$ nodes transforms [18] to the $P3|fix_j|C_{max}$ problem that is strongly NP-hard [20]. On the other hand, the SA subproblem defined on *the clockwise direction of a bidirectional ring* is a special case of the unidirectional ring problem inasmuch as its input consists of only the subset of demands that are routed in that direction. Therefore, the problem can be solved in polynomial time for small rings, and approximation algorithms with constant ratios exist, as we show next.

Since any algorithm that solves the $P|fix_j|C_{max}$ problem also solves the SA problem, in the following we will derive results for the SA problem in bidirectional rings by studying the corresponding multiprocessor scheduling problem. In our discussion, we will make use of two concepts related to $P|fix_j|C_{max}$.

Definition III.1 (Compatible Tasks): A set $\mathcal{T}$ of tasks for the $P|fix_j|C_{max}$ problem are said to be *compatible* if and only if their prespecified sets of processors are pairwise disjoint, i.e., $fix_i \cap fix_j = \emptyset, \forall i, j \in \mathcal{T}$.

Compatible tasks may be paired with each other (i.e., they can be executed simultaneously), as they do not share any processors.

Definition III.2 (Dominant Processor and Lower Bound): Consider an instance of $P|fix_j|C_{max}$, and let $\mathcal{T}_k$ denote the set of tasks that require processor k, i.e., $\mathcal{T}_k = \{j : k \in fix_j\}$. Clearly, all the tasks in $\mathcal{T}_k$ are pairwise incompatible, hence they have to be executed sequentially. Let Π_k denote the sum of processing times of tasks that require processor k:

$$\Pi_k = \sum_{j \in \mathcal{T}_k} p_j, k = 1, \ldots, m. \quad (1)$$

Then, a lower bound LB for the problem instance can be obtained as:

$$LB = \max_{k=1,\ldots,m} \{\Pi_k\}. \quad (2)$$

We will refer to a processor that achieves the lower bound LB as the *dominant* processor.

A. Complexity Results for Rings With $N = 3, 4$ Nodes

The following two lemmas establish that, under shortest path routing, the SA problem can be solved in polynomial time in three- and four-node bidirectional rings, since the subproblems defined on the clockwise (and, hence, also the counter-clockwise) direction yield polynomial solutions. Note also that, in the special case whereby all demands are equal to one slot, $t_{sd} = 1$, the spectrum contiguity constraint becomes redundant, and the SA problem reduces to the wavelength assignment (WA) problem [27]. Consequently, these two lemmas also establish that the WA problem is solvable in polynomial time in three- and four-node rings with shortest path routing.

Lemma III.1: The SA subproblem defined in the clockwise direction of a bidirectional ring with $N = 3$ nodes and shortest path routing is solvable in polynomial time.

Proof: In a bidirectional ring with $N = 3$ nodes, the shortest path for each demand consists of a single link. Consider the SA subproblem defined on the clockwise direction. This subproblem has three demands, each carried on exactly one of the three clockwise links of the ring. The corresponding $P3|fix_j|C_{max}$ multiprocessor scheduling problem has three tasks, each requiring exactly one of the three processors (i.e., $|fix_j| = 1, j = 1, 2, 3$). Since the tasks are pairwise compatible, they can be scheduled simultaneously. Hence, the optimal value of the total amount of spectrum required in the network (respectively, C_{max}) is equal to the maximum demand size (respectively, the maximum task processing time). ■

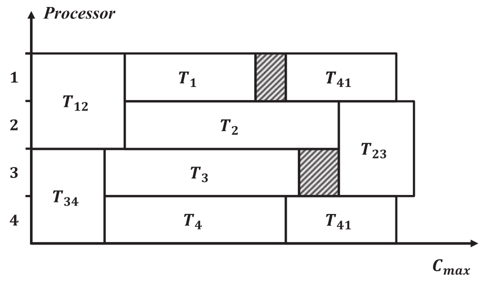

Fig. 2. Optimal schedule for the clockwise direction of a four-node bidirectional ring with shortest path routing.

Lemma III.2: The SA subproblem defined in the clockwise direction of a bidirectional ring with $N = 4$ nodes and shortest path routing is solvable in polynomial time.

Proof: In a four-node ring, the clockwise and counter-clockwise paths between two non-adjacent nodes are of equal length (i.e., two), and either may be selected as the shortest path. Let us consider the case where all demands between non-adjacent nodes are routed in the clockwise direction. In other words, for non-adjacent nodes 1 and 3, both traffic from 1 to 3 and traffic from 3 to 1 is routed clockwise; and similarly for the other pair (2,4) of non-adjacent nodes. Hence, the input to the SA subproblem consists of four one-link demands and four two-link demands. Consequently, the input to the corresponding $P4|fix_j|C_{max}$ problem consists of four single-processor tasks and four two-processor tasks. Let us denote these tasks as T_1, T_2, T_3, T_4, T_{12}, T_{23}, T_{34}, and T_{41}, where the subscript of each task denotes the processors in the corresponding set fix_j.

The proof is by construction of the optimal schedule, as shown in Fig. 2. Specifically, first schedule the task T_{12} in parallel with the task T_{34} starting at time $t = 0$. Then, add all the single processor tasks T_1, T_2, T_3, T_4 to this initial schedule without any gaps. Finally, execute the two-processor tasks T_{23} and T_{41} as soon as both processors of each task are available. For the instance depicted in Fig. 2, the schedule is optimal as it is equal to the lower bound determined by the sum of the processing times of tasks requiring processor 2 (the dominant processor). In fact, because of symmetry, the schedule is optimal regardless of which processor is the dominant one.

If some of the demands between non-adjacent nodes are routed in the counter-clockwise direction, then the instance of $P4|fix_j|C_{max}$ defined on the clockwise direction will not include the corresponding two-processor tasks. Again, it can be seen that the above algorithm yields an optimal schedule. For instance, if task T_{23} is excluded from Fig. 2, then the schedule remains optimal. The same is true if T_{41} is excluded, or T_{23} and T_{41} are both excluded, or any combination of two-processor tasks is excluded. If all two-processor tasks are excluded (i.e., all demands between non-adjacent nodes are routed in the counter-clockwise direction), then the problem contains only single-processor tasks and the algorithm again produces an optimal schedule. ■

The above lemma shows that as long as traffic demands in a four-node bidirectional ring are routed along a shortest path (with ties broken arbitrarily), the SA problem is solvable in polynomial time using a simple algorithm that is linear in the number of tasks (spectrum demands). The following lemma shows that if one of the demands between adjacent nodes takes a non-shortest path, the SA problem becomes NP-complete. The proof is by reduction from the PARTITION problem [28] which is defined as:

Definition III.3 (PARTITION): Given a set of k integers $A = \{a_1, a_2, \ldots, a_k\}$ such that $B = \sum_{j=1}^{k} a_j$, does there exist a partition of A into two sets, A_1 and A_2, such that $\sum_{a_j \in A_1} a_j = \sum_{a_j \in A_2} a_j = \frac{B}{2}$?

Following standard NP-Completeness proofs, the proof of the following lemma, as well as that of Theorem III.1, shows that (1) there is a polynomial transformation of any instance of PARTITION to an instance of the corresponding $Pm|fix_j|C_{max}$ problem, and (2) a partition exists if and only if an optimal schedule for the $Pm|fix_j|C_{max}$ problem also exists. Specifically, in both proofs, we include a number of gadget tasks in the instance of the $Pm|fix_j|C_{max}$ problem that are independent of the PARTITION instance, and select the C_{max} value to ensure that the second condition above is satisfied.

Lemma III.3: The SA subproblem defined in the clockwise direction of a bidirectional ring with $N = 4$ nodes and such that:

- all demands between non-adjacent nodes are routed in the clockwise direction, and
- all demands between adjacent nodes are routed along their (one-link) shortest path in the clockwise or counter-clockwise direction, *except* for one such demand that is directed to a three-link path in the clockwise direction,

is NP-complete.

Proof: If a traffic demand with a one-link shortest path in the counter-clockwise direction is routed along the alternate clockwise three-link path, then the $P4|fix_j|C_{max}$ problem defined on the clockwise direction will include a three-processor task. Without loss of generality, assume that this three-processor task requires processors 3, 4, and 1 (similar arguments apply for any other three-processor task). Given an instance of PARTITION, we create an instance of this $P_4|fix_j|C_{max}$ as follows. For each $a_j \in A$ we create a task τ_j with processing time $p_j = a_j$ and $fix_j = \{2\}$ (note that these tasks must be executed by processor 2, the one that is *not* required by the three-processor task). We also create the following eight gadget tasks:

task	p_j	fix_j
T_a	B	$\{1,2\}$
T_b	$B/2$	$\{3,4\}$
T_c	B	$\{2,3\}$
T_d	$B/2$	$\{4,1\}$
T_e	$B/2$	$\{3,4,1\}$
T_1	B	$\{1\}$
T_2	B	$\{3\}$
T_3	$3B/2$	$\{4\}.$

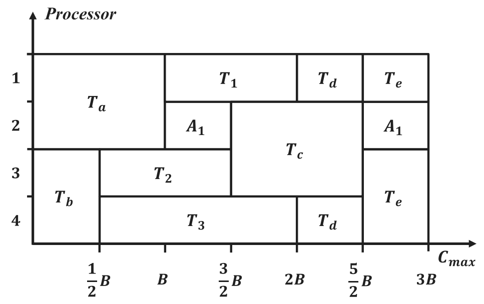

Fig. 3. A feasible schedule with $C_{max} = 3B$ for the clockwise direction of a four-node bidirectional ring with shortest-path routing except for one demand routed along a three-link path.

If A can be partitioned into two disjoint sets A_1 and A_2 such that $\sum_{a_j \in A_1} = \sum_{a_j \in A_2} = B/2$, then there is a feasible schedule with $C_{max} = 3B$, as shown in Fig. 3.

Conversely, let us assume that there exists a feasible schedule $\mathcal{S}$ with $C_{max} \leq 3B$. Without loss of generality, suppose that T_a and T_b are executed before T_c and T_d in $\mathcal{S}$; otherwise, we can use similar arguments and reach the same conclusion. Then, all the single processor tasks T_1, T_2, and T_3 must be executed immediately after T_a or T_b complete, as scheduling any other task at that time would lead to a makespan greater than $3B$. T_c must also be scheduled exactly right after T_2 and before T_e, otherwise it would not be possible to obtain the schedule with length of at most $3B$. Using a similar argument, T_d must be scheduled right after T_1 and T_3 and before T_e, and in parallel with T_c. The schedule corresponding to this set of tasks is shown in Fig. 3 where only the intervals $[B, 3B/2]$ and $[5B/2, 3B]$ are available for the execution of the PARTITION jobs on processor 2. Therefore, a partition must exist. ■

B. Complexity Results for Rings With $N \geq 5$ Nodes

The next theorem states that the SA problem on five-node bidirectional rings (and, hence, on any larger ring) is intractable.

Theorem III.1: The SA subproblem defined in the clockwise direction of a bidirectional rings with $N = 5$ nodes and shortest path routing is NP-complete.

Proof: As the number of nodes is odd, there is a unique shortest path for each traffic demand between any two non-adjacent nodes; therefore, the problem in the clockwise direction includes only the demands with a shortest path along the clockwise links. The proof is by reduction from the PARTITION problem, and follows an approach similar to the one we used in the proof of Lemma III.3. Specifically, for each $a_j \in A$, we create a task τ_j

task	p_j	fix_j
T_a	$3B/2$	$\{1,2\}$
T_b	$5B/2$	$\{2,3\}$
T_c	$B/2$	$\{3,4\}$
T_d	B	$\{4,5\}$
T_e	$2B$	$\{5,1\}$
T_1	$3B/2$	$\{1\}$
T_2	$2B$	$\{3\}$
T_3	$7B/2$	$\{4\}$
T_4	$2B$	$\{5\}.$

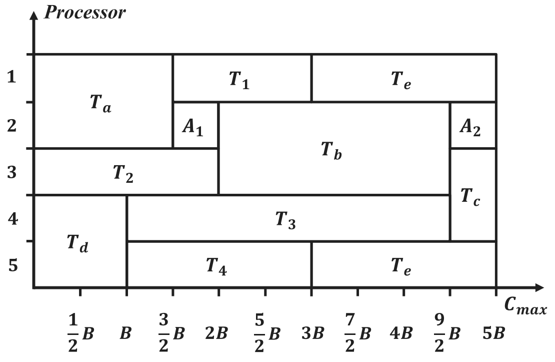
Processor
1
2
3
4
5
T_a
T_1
T_e
A_1
T_b
A_2
T_2
T_c
T_3
T_d
T_4
T_e
C_{max}
$\frac{1}{2}B$ B $\frac{3}{2}B$ $2B$ $\frac{5}{2}B$ $3B$ $\frac{7}{2}B$ $4B$ $\frac{9}{2}B$ $5B$

Fig. 4. A feasible schedule with $C_{max} = 5B$ for the clockwise direction of a five-node ring with shortest path routing.

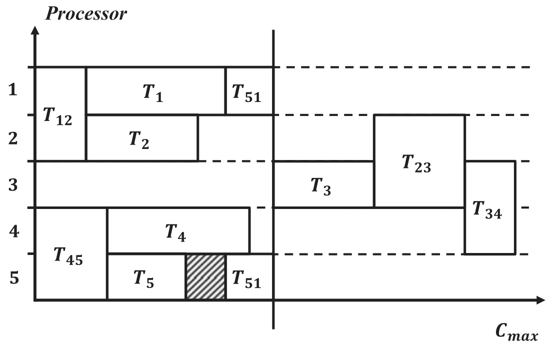

Fig. 5. Two-part schedule for a five-node bidirectional ring with shortest path routing.

with processing time $p_j = a_j$ and $fix_j = \{2\}$. We also create the following set of tasks:

If there exists a partition of A into two disjoint sets A_1 and A_2 such that $\sum_{a_j \in A_1} = \sum_{a_j \in A_2} = B/2$, then we can execute the tasks as shown in Fig. 4 and create a feasible schedule with $C_{max} = 5B$.

Conversely, assume that there exists a feasible schedule $\mathcal{S}$ with $C_{max} \leq 5B$. Similar to the proof of Lemma III.3 and without loss of generality, suppose that T_a and T_d are executed before T_c and T_e in $\mathcal{S}$; otherwise, we can use similar arguments and reach the same conclusion. We need to schedule T_2 in parallel with T_a and T_d, otherwise the schedule length will exceed $5B$. As T_d completes earlier than T_a, we need to execute T_4 before T_e. Therefore, T_1 must be scheduled right after T_a and before T_e. On the other hand, T_3 must be executed immediately after T_d, and T_c must be scheduled at the very end of $\mathcal{S}$, since if we change the order of execution of T_3 and T_c in $\mathcal{S}$, the makespan will be greater than $5B$. Finally, executing T_c between $[9B/2, 5B]$ means that T_b must be scheduled immediately after T_2. A feasible schedule corresponding to this set of tasks is shown in Fig. 4 where only the intervals $[3B/2, 2B]$ and $[9B/2, 5B]$ are available for the execution of the PARTITION jobs on processor 2. Thus, we conclude that a partition of A must exist. ■

C. Approximation Algorithms

In this section, we first provide approximation algorithms for the SA problem on bidirectional rings with $N = 5, 6$ and 7 nodes under shortest path routing. We then develop approximation algorithms for bidirectional rings with $N \geq 8$ nodes. Since, as we mentioned earlier, the WA problem is a special case of SA, all approximation algorithms in this section also apply to WA.

1) Rings With $N = 5 - 7$ Nodes:

Lemma III.4: There exists an 1.5-approximation algorithm for the SA subproblem defined on the clockwise direction of a bidirectional ring with $N = 5$ nodes and shortest path routing.

Proof: As we mentioned earlier, in a five-node ring each traffic demand has a unique shortest path. Therefore, the clockwise direction serves $10 (= 5 \times 4/2)$ demands, and the corresponding scheduling problem has 10 tasks as shown in Fig. 5, where the subscript of each task indicates the processors required by the task. Without loss of generality, let processor 3 be the dominant processor, i.e., the one that achieves the lower bound LB in (2). Let OPT denote the optimal value of the makespan for this problem; clearly, $LB \leq OPT$.

Consider now the seven tasks that do *not* require processor 3, shown in the left part of the schedule in Fig. 5. The scheduling problem consisting of these seven tasks can be viewed as the scheduling problem on a four-processor system (i.e., one without processor 3), similar to the one depicted in Fig. 3—but with three rather than four two-processor tasks. In essence, this scheduling problem corresponds to the SA problem on the clockwise direction of the five-node after removing the link corresponding to processor 3 and the three traffic demands using that link. Based on our earlier result regarding the four-node rings, these seven tasks can be scheduled optimally, as shown on the left part of Fig. 5. Let OPT' be the makespan of this schedule; then, $OPT' \leq OPT$.

Now consider the three tasks that require processor 3. These can be scheduled back-to-back without any gaps, as shown in the right part of Fig. 5. The makespan of this schedule is equal to LB. Hence, the makespan of the two-part, ten-task schedule depicted in Fig. 5 is equal to: $OPT' + LB \leq 2 \times OPT$.

We can improve the approximation ratio of 2 by modifying the above two-part schedule as follows. Without loss of generality, assume that $T_{23} \geq T_{34}$ as indicated in Fig. 5; if T_{34} is larger than T_{23}, then simply reverse the roles in the following discussion. In this case, we have that:

$$\begin{aligned} T_{34} &\leq T_3 + T_{23} \\ \Rightarrow 2T_{34} &\leq T_3 + T_{23} + T_{34} = LB \leq OPT \\ \Rightarrow T_{34} &\leq 0.5 \times OPT. \end{aligned} \quad (3)$$

Now slide the right part of the schedule in Fig. 5 (i.e., the three tasks T_3, T_{23} and T_{34}) as far left as possible so that tasks T_3 and/or T_{23} overlap with the tasks in the left part of the schedule. Consider the resulting nine-task schedule, i.e., the one consisting of all tasks of the problem except T_{34}. It can be seen that this schedule is optimal for these nine tasks. Let OPT'' be the makespan of this nine-task schedule, and $OPT'' \leq OPT$. Scheduling task T_{34} immediately after the end of this schedule results in a ten-task schedule of length $OPT'' + T_{34}$. Using (3), we conclude that the makespan of this schedule is no larger than $1.5 \times OPT$. ■

Lemma III.5: There exist 2-approximation algorithms for the SA subproblem defined on the clockwise direction of bidirectional rings with $N = 6, 7$ nodes and shortest path routing.

Proof: The proof is by construction of a two-part schedule similar to the one we created for the proof of Lemma III.4. The proof is omitted due to its length, and the details are available in the first author's dissertation. ■

2) Rings With $N \geq 8$ Nodes: We now present a general approximation algorithm for rings of any size. Consider the SA problem defined on the clockwise direction of a ring with $N \geq 8$ nodes and shortest path routing. The key idea is based on the observation that if we remove a link from the ring along with the traffic demands whose shortest paths use this link, the resulting SA subproblem is equivalent to the SA problem on a directed path with $N - 1$ nodes. Furthermore, the $2 + \epsilon$ approximation algorithm in [29] for computing the interval chromatic number of interval graphs can be used to solve the SA problem in chain networks with the same performance bound [17]. Therefore, the approximation algorithm for rings consists of the following steps:

1) Formulate the $PN|fix_j|C_{max}$ problem for the clockwise direction of the original ring.
2) Let processor N be the dominant processor (and relabel the processors appropriately if necessary).
3) Remove processor N and all tasks j that use this processor (i.e., tasks j such that $N \in fix_j$); the resulting scheduling problem corresponds to the SA problem on a $(N - 1)$-link chain.
4) Use the $2 + \epsilon$ approximation algorithm in [29] to create schedule $\mathcal{S}_1$ for the scheduling problem on the chain network.
5) Schedule all tasks that use processor N sequentially without any gaps to create schedule $\mathcal{S}_2$.
6) Concatenate schedules $\mathcal{S}_1$ and $\mathcal{S}_2$ to create schedule $\mathcal{S}$ for the ring network.

Let OPT be the optimal makespan for the ring network. By construction, the makespan of $\mathcal{S}_2$ is equal to $LB \leq OPT$, while the makespan of $\mathcal{S}_1$ is no longer than $(2 + \epsilon)OPT$. Hence, the approximation ratio of the above algorithm for an N-node ring is $3 + \epsilon$, better than the $4 + 2\epsilon$ algorithm presented in [17].

D. Evaluation

The approximation ratios of the algorithms described in the previous subsection correspond to worst-case inputs, and we expect that the algorithms will perform better on average. To investigate the average-case performance of the algorithms, we have carried out simulation experiments on rings of various sizes. We assume that the network supports the following data rates (in Gbps): 10, 40, 100, 400, and 1000. For each problem instance, we generate random traffic rates between every pair of nodes based on one of three distributions: (1) *Uniform*: traffic demands may take any of the five discrete values in the set $\{10, 40, 100, 400, 1000\}$ with equal probability; (2) *Skewed low*: traffic demands may take one of the five discrete values above with probabilities $0.30, 0.25, 0.20, 0.15$, and 0.10, respectively (i.e., the lower data rates have higher probability to be selected); or (3) *Skewed high*: traffic demands may take one of the five discrete values above with probabilities $0.10, 0.15, 0.20, 0.25$, and 0.30, respectively (i.e., the higher data rates have higher probability to be selected). Once the traffic rates between every source-destination pair have been generated, we calculate the corresponding spectrum slots as follows. We assume that the slot width is 12.5 GHz, and the 16-QAM modulation format, such that demands of size 10, 40, 100, 400, and 1000 Gbps require 1, 1, 2, 8, and 20 slots, respectively, consistent with the values used in [30, Table 1].

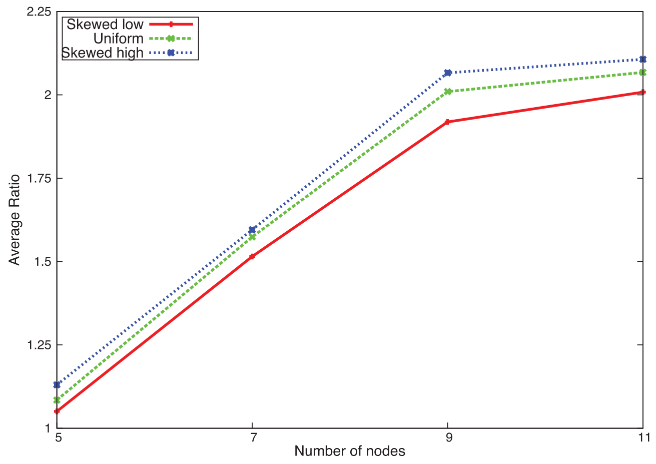

Fig. 6. Average ratio of solutions produced by the approximation algorithms to the lower bound.

Since the optimal solution is not known for rings with more than four nodes, we compute the lower bound as in expression (2). We then compute the ratio of the makespan produced by the algorithm to the lower bound. Note that the lower bound is not tight, as it ignores any gaps introduced by the scheduling of incompatible tasks in the optimal solution. Therefore, this ratio overestimates the difference between the solution produced by the algorithm and the optimal one. Fig. 6 plots this ratio as a function of ring size for the three demand distributions; each data point in the figure represents the average of thirty random problem instances.

As we can see, the average performance of the approximation algorithms is significantly better than what their respective constant (worst-case) ratios suggest. For instance, for a five-node ring, the algorithm is within 15% of the lower bound although its worst-case ratio is 1.5; whereas for a seven-node ring, the worst-case ratio is 2, but the average ratio is at most 1.6. Further, for rings of nine or more nodes, the worst-case ratio is 3, but the average ratio is around 2. Recall that the average ratio is relative to the lower bound, not the optimal, hence the actual performance of the algorithms (i.e., compared to the optimal solution) is better than the figure suggests.

Finally, we note that for the problem instances used to derive the results of Fig. 6, the running time of the approximation algorithms was about 15 ms on a 3.10 GHz 4-core Xeon CPU; this value did not depend on the demand distributions or ring sizes used in our experiments.

IV. RSA IN RINGS

Let us now turn our attention to the RSA problem in bidirectional rings. Unlike the previous section where we assumed that traffic demands are routed on the shortest path, our objective is to determine both a route and a spectrum allocation for each demand. Since there are exactly two paths between each pair of nodes in a ring network, the general RSA problem on rings reduces to the RSA problem with $k = 2$ fixed-alternate paths in Definition II.1.

We first present results to establish the complexity of the RSA problem in rings, followed by a new approximation algorithm. Our discussion builds upon the results of Lemma II.1 which shows that the RSA problem with fixed-alternate routing is a special case of the $P|set_j|C_{max}$ multiprocessor scheduling problem.

A. Complexity Results

Lemma IV.1: The RSA problem in 3-node bidirectional rings is solvable optimally in polynomial time.

Proof: We will show that in a bidirectional ring with $N = 3$ nodes, the solution in which each traffic demand takes the shortest (i.e., one-link) path, is optimal.

Consider the corresponding $P|set_j|C_{max}$ with six processors and six tasks. Clearly, the length of the longest task is a lower bound on the optimal makespan, i.e., $OPT \geq LB = \max_{j=1,\ldots,6}\{p_j\}$. In the solution to the $P|set_j|C_{max}$ problem defined by shortest path routing in the RSA instance, each task is executed on a different single processor. Hence, the tasks are pairwise compatible and may all start execution at time $t = 0$. Consequently, this solution is optimal as its makespan is equal to $\max_{j=1,\ldots,6}\{p_j\}$. ■

Theorem IV.1: The RSA problem in 4-node bidirectional rings is NP-Complete.

Proof: Consider a 4-node bidirectional ring with traffic demands between each pair of nodes, for a total of 12 $(= 4 \times 3)$ types of demands. Let $\{1, 2, 3, 4, 1', 2', 3', 4'\}$, denote the eight directed links of the network such that l and $l', l = 1, 2, 3, 4$, are the links in the clockwise and counter-clockwise direction, respectively, between adjacent nodes in the ring. Also, assume that, in the clockwise direction, link 1 is adjacent to 2, 2 is adjacent to 3, 3 to 4, and 4 to 1, and similarly for links in the counter-clockwise direction. Each demand may be assigned a path in either the clockwise or counter-clockwise direction. For instance, a demand may take either a one-link path (say, along link 1) in the clockwise direction or the three-link path (along links $4'$, $3'$, and $2'$) in the counter-clockwise direction.

The equivalent multiprocessor scheduling problem $P8|set_j|C_{max}$ has $m = 8$ processors, which we assume are labeled identically to the corresponding links, and is constructed according to Lemma II.1. We will prove that this scheduling problem is NP-Complete by reduction from the PARTITION problem.

Given an instance of PARTITION, we create an instance of $P8|set_j|C_{max}$ with the eight processors $\{1, 2, 3, 4, 1', 2', 3', 4'\}$. For each $a_j \in A$, we create a task τ_j with processing time $p_j = a_j$ and $set_j = \{\{2\}, \{1', 4', 3'\}\}$; in the equivalent RSA problem, this demand may be routed either along link 2 in the clockwise direction or along the path

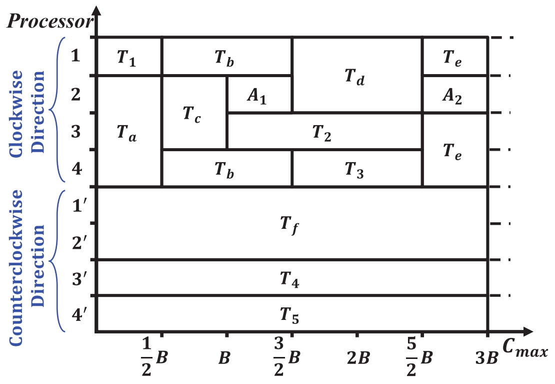

Fig. 7. A feasible schedule for the RSA problem in a four-node bidirectional ring with $C_{max} = 3B$.

$< 1', 4', 3' >$ in the counter-clockwise direction. We also create the following eleven gadget tasks:

task j	p_j	set_j
T_a	$B/2$	$\{\{2, 3, 4\}, \{1'\}\}$
T_b	B	$\{\{4, 1\}, \{3', 2'\}\}$
T_c	$B/2$	$\{\{2, 3\}, \{1', 4'\}\}$
T_d	B	$\{\{1, 2\}, \{4', 3'\}\}$
T_e	$B/2$	$\{\{3, 4, 1\}, \{2'\}\}$
T_f	$3B$	$\{\{3, 4\}, \{2', 1'\}\}$
T_1	$B/2$	$\{\{1\}, \{4', 3', 2'\}\}$
T_2	$3B/2$	$\{\{3\}, \{2', 1', 4'\}\}$
T_3	B	$\{\{4\}, \{3', 2', 1'\}\}$
T_4	$3B$	$\{\{4, 1, 2\}, \{3'\}\}$
T_5	$3B$	$\{\{1, 2, 3\}, \{4'\}\}$.

If it is possible to partition A into A_1 and A_2 such that $\sum_{a_j \in A_1} = \sum_{a_j \in A_2} = B/2$, then there exists a feasible schedule as shown in Fig. 7 with $C_{max} = 3B$.

Conversely, suppose that there exists a feasible schedule $\mathcal{S}$ with $C_{max} \leq 3B$. Since T_4 and T_5 have length equal to $3B$, they must be executed on their respective single-processor set; scheduling either of them on the respective three-processor set would create conflict with some other task, resulting in a longer schedule. T_f, also of length $3B$, must be executed on its two-processor set that is compatible with the single-processor sets of T_4 and T_5. Since all four processors $1', 2', 3'$, and $4'$ are busy in the interval $[0, 3B]$ (equivalently, the counter-clockwise direction of the ring is fully utilized), the remaining tasks must be assigned to the other four processors (equivalently, the corresponding demands must be routed in the clockwise direction) to ensure that $C_{max} \leq 3B$.

Without loss of generality, assume that T_a is executed before T_e in $\mathcal{S}$; otherwise, similar arguments can be used to reach the same conclusion. T_1 is the only remaining task that is compatible with T_a and must be executed in parallel with the latter. Then, tasks T_b and T_c must be scheduled immediately after T_a and T_1 complete. Since T_c completes earlier than T_b, T_2 must be executed immediately following T_c, otherwise the schedule length for the clockwise direction would exceed $3B$. Similarly, as soon as T_b completes execution, T_d and T_3 must be scheduled in parallel. Finally, we note that the only remaining gadget task, T_e, must be appended at the end of this schedule of tasks to ensure that the makespan does not exceed $3B$. The schedule corresponding to this ordering of tasks is shown in Fig. 7, where only the intervals $[B, 3B/2]$ and $[5/2B, 3B]$ are available for the PARTITION jobs. Thus, a partition exists. ■

B. Approximation Algorithms

The best approximation algorithm for the m-processor scheduling problem $Pm|set_j|C_{max}$ was developed in [22] and has a ratio of $m/2$. The algorithm proceeds in two phases:

1) *Processor Assignment:* In the first phase, each task j is assigned to one of its alternate processor sets in set_j. Consider an assignment $\mathcal{F}$, and let $LB_{\mathcal{F}}$ denote the processing time on the dominant processor under $\mathcal{F}$, as given by expression (2). A dynamic programming algorithm was developed in [22] to obtain in pseudopolynomial time an optimal assignment $\mathcal{F}^\star$ such that $LB^\star = LB_{\mathcal{F}^\star}$ is minimum over all possible assignments. Clearly, $LB^\star$ is a lower bound on the optimal makespan OPT for the original $Pm|set_j|C_{max}$ problem, i.e., $LB^\star \leq OPT$.
2) *Task Scheduling:* Given the assignment $\mathcal{F}^\star$, the original problem reduces to a $Pm|fix_j|C_{max}$ problem in which the objective is to schedule the tasks so as to minimize the makespan. A polynomial heuristic is used to solve this problem, and it is shown that the makespan C achieved by this scheduling heuristic is such that $C \leq (m/2)LB^\star$. Hence, the two-phase algorithm is an $(m/2)$-approximation algorithm for the original $Pm|set_j|C_{max}$ problem.

The above two-phase approximation algorithm for $Pm|set_j|C_{max}$ corresponds to a natural decomposition of the RSA problem into two subproblems that are solved sequentially: a *routing* problem (in which a demand is assigned to either the clockwise or counter-clockwise path in a manner that takes into account the spectrum demands), and a *SA* problem (in which spectrum is assigned to each demand along the path determined by the solution to the routing problem).

Note that a ring with N nodes has a total of $m = 2N$ directed links (i.e., processors in the corresponding scheduling problem). Hence, a straightforward application of the two-phase approximation algorithm to the RSA problem in rings would yield an approximation ratio of N. However, as we noted earlier, once the routing of demands has been determined, the clockwise and counter-clockwise directions of the ring become independent of each other and the corresponding SA problems may be solved separately. Therefore, rather than solving a single $P2N|fix_j|C_{max}$ problem in the second phase, it is only necessary to solve two $PN|fix_j|C_{max}$ problems, one for each direction of the ring. With this observation, the approximation ratio of the two-phase algorithm for the RSA problem in rings is $N/2$ rather than N.

We now show that it is possible to further improve the approximation ratio for the RSA problem in rings. First, we note that linear time approximation ratios for the $P4|fix_j|C_{max}$ and $P5|fix_j|C_{max}$ problems with ratios of 1.5 and 2, respectively, were developed in [31]. By using these algorithms in the task scheduling phase above, rather than the general one presented in [22], the two-phase algorithm yields approximation ratios of 1.5 and 2 for rings with $N = 4$ and $N = 5$ nodes, respectively.

For larger rings (i.e., $N \geq 6$), we leverage the approximation algorithm for the SA problem we developed in Section III-C2 to obtain a two-phase approximation algorithm for the RSA problem with a constant ratio that is smaller than $N/2$:

1) *Routing:* Use the dynamic programming algorithm in [22] to assign each traffic demand to the clockwise or counter-clockwise path.
2) *Spectrum Assignment:* Consider only the traffic demands routed along the clockwise direction and assign spectrum to them by solving the corresponding $PN|fix_j|C_{max}$ problem with the approximation algorithm in Section III-C2; repeat for the traffic demands in the counter-clockwise direction.

Following similar arguments as in Section III-C2, we conclude that the above two-phase algorithm for the RSA problem in N-node rings has an approximation ratio of $3 + \epsilon$.

V. Concluding Remarks

We have studied the complexity of the RSA problem in bidirectional rings and we have developed new constant-ratio approximation algorithms. In future work, we plan to apply multiprocessor scheduling theory to tackle the distance-adaptive RSA problem in ring and mesh networks.

References

[1] O. Gerstel, M. Jinno, A. Lord, and S. J B Yoo, "Elastic optical networking: A new dawn for the optical layer?" *IEEE Commun. Mag.*, vol. 50, no. 2, pp. s12–s20, Feb. 2012.

[2] M. Jinno, H. Takara, and B. Kozicki, "Dynamic optical mesh networks: Drivers, challenges and solutions for the future," in *Proc. 35th Eur. Conf. Opt. Commun.*, Sep. 2009, pp. 1–4.

[3] G. Shen and M. Zukerman, "Spectrum-efficient and agile CO-OFDM optical transport networks: architecture, design, and operation," *IEEE Commun. Mag.*, vol. 50, no. 5, pp. 82–89, May 2012.

[4] G. Zhang, M. De Leenheer, A. Morea, and B. Mukherjee, "A survey on OFDM-based elastic core optical networking," *IEEE Commun. Surveys Tut.*, vol. 15, no. 1, pp. 65–87, Jan.–Apr. 2013.

[5] K. Christodoulopoulos, I. Tomkos, and E. A. Varvarigos, "Elastic bandwidth allocation in flexible OFDM–based optical networks," *J. Lightw. Technol.*, vol. 29, no. 9, pp. 1354–1366, Aug. 2011.

[6] M. Klinkowski and K. Walkowiak, "Routing and spectrum assignment in spectrum sliced elastic optical path network," *IEEE Commun. Lett.*, vol. 15, no. 8, pp. 884–886, May 2011.

[7] L. Velasco, M. Klinkowski, M. Ruiz, and J. Comellas, "Modeling the routing and spectrum allocation problem for flexgrid optical networks," *Photon. Netw. Commun.*, vol. 24, pp. 177–186, 2012.

[8] S. Talebi, F. Alam, I. Katib, M. Khamis, R. Khalifah, and G. N. Rouskas, "Spectrum management techniques for elastic optical networks: A survey," *Opt. Switching Netw.*, vol. 13, pp. 34–48, Jul. 2014.

[9] I.-F. Chao and M. C. Yuang, "Toward wireless backhaul using circuit emulation over optical packet-switched metro WDM ring network," *J. Lightw. Technol.*, vol. 31, no. 18, pp. 3032–3042, Sep. 2013.

[10] C. Cao, H. Fu, J. Wang, X. Tan, and Y. Zhang, "Round robin ring for metro wireless backhaul networks," presented at the *Asia Commun. Photon. Conf.*, Beijing, China, Nov. 2013.

[11] Q. Wei, J. Bazzi, M. Lott, and Y. Pointurier, "Multicast in mobile backhaul with optical packet ring," presented at the *6th Int. Workshop Sel. Topics Mobile Wireless Comput.*, Lyon, France, Oct. 2013.

[12] *International Telecommunications Union. Ethernet Ring Protection Switching*, ITU-T Standard G.8032, Feb. 2012.

[13] F. Musumeci, F. F. Puleio, and M. Tornatore, "Dynamic grooming and spectrum allocation in optical metro ring networks with flexible grid," presented at the *Int. Conf. Transparent Opt. Netw.*, Jun. 2013, Paper We.A1.2.

[14] C. Rottondi, M. Tornatore, A. Pattavina, and G. Gavioli, "Routing, modulation level, and spectrum assignment in optical metro ring networks using elastic transceivers," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 5, no. 4, pp. 305–315, Apr. 2013.

[15] Y. Wang, X. Cao, and Y. Pan, "A study of the routing and spectrum allocation in spectrum-sliced elastic optical path networks," in *Proc. IEEE INFOCOM*, 2011, pp. 1503–1511.

[16] I. Popescu, I. Cerutti, N. Sambo, and P. Castoldi, "On the optimal design of a spectrum-switched optical network with multiple modulation formats and rates," *IEEE/OSA J. Opt. Commun. Netw.*, vol. 5, no. 11, pp. 1275–1284, Nov. 2013.

[17] S. Shirazipourazad, C. Zhou, Z. Derakhshandeh, and A. Sen, "On routing and spectrum allocation in spectrum-sliced optical networks," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 385–389.

[18] S. Talebi, E. Bampis, G. Lucarelli, I. Katib, and G. N. Rouskas, "The spectrum assignment (SA) problem in optical networks: A multiprocessor scheduling perspective," in *Proc. Opt. Netw. Des. Model.*, May 2014, pp. 55–60.

[19] E. Bampis, M. Caramia, J. Fiala, A. Fishkin, and A. Iovanella, "Scheduling of independent dedicated multiprocessor tasks," in *Proc. 13th Annu. Int. Symp. Algorithms Comput.*, 2002, vol. 2518, pp. 391–402.

[20] J. A. Hoogeveen, S. L. Van de Velde, and B. Veltman, "Complexity of scheduling multiprocessor tasks with prespecified processor allocations," *Discree Appl. Math.*, vol. 55, pp. 259–272, 1994.

[21] L. Bianco, J. Blazewicz, P. Dell'Olmo, and M. Drozdowski, "Scheduling multiprocessor tasks on a dynamic configuration of dedicated processors," *Ann. Oper. Res.*, vol. 58, no. 7, pp. 493–517, 1995.

[22] J. Chen and Ch. Lee, "General multiprocessor task scheduling," *Nav. Res. Logist.*, vol. 46, no. 1, pp. 57–74, 1999.

[23] K. Jansen and L. Porkolab, "General multiprocessor task scheduling: Approximate solutions in linear time," *SIAM J. Comput.*, vol. 35, no. 3, pp. 519–530, 2005.

[24] L. Torres A. Miranda, and J. Chen, "On the approximability of multiprocessor task scheduling problems," in *Proc. 13th Annu. Int. Symp. Algorithms Comput.*, 2002, vol. 2518, pp. 403–415.

[25] M. Kubal, "The complexity of scheduling independent two-processor tasks on dedicated processors," *Inform. Process. Lett.*, vol. 24, no. 3, pp. 141–147, 1987.

[26] J. Chen and A. Miranda, "A polynomial time approximation scheme for general multiprocessor job scheduling," *SIAM J. Comput.*, vol. 31, no. 1, pp. 1–17, 2001.

[27] G. N. Rouskas, "Routing and wavelength assignment in optical WDM networks," in *Wiley Encyclopedia of Telecommunications*, J. Proakis, Ed. New York, NY, USA: Wiley, 2001.

[28] M. R. Garey and D. S. Johnson, *Computers and Intractability*. New York, NY, USA: Freeman, 1979.

[29] A. L. Buchsbaum, H. Karloff, C. Kenyon, N. Reingold, and M. Thorup, "Opt versus load in dynamic storage allocation," *SIAM J. Comput.*, vol. 33, no. 3, pp. 632–646, 2004.

[30] M. Jinno, B. Kozicki, H. Takara, A. Watanabe, Y. Sone, T. Tanaka, and A. Hirano, "Distance-adaptive spectrum resource allocation in spectrum-sliced elastic optical path network," *IEEE Commun. Mag.*, vol. 48, no. 8, pp. 138–145, Aug. 2010.

[31] J. Huang, J. Chen, S. Chen, and J. Wang, "A simple linear time approximation algorithm for multi-processor job scheduling on four processors," *J. Combinatorial Optim.*, vol. 13, pp. 33–45, 2007.

Authors' biographies not available at the time of publication.